Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

Software Reliability Prediction : A Review

Suneel Kumar Rath
C. V. RAMAN Global University
Bhubaneswar, Odisha, India

Madhusmita Sahu
C. V. RAMAN Global University
Bhubaneswar, Odisha, India

Shom Prasad Das
Birla Global University
Bhubaneswar, Odisha, India

***Abstract*—Software Reliability is the quality of a system that performs its required operations under static conditions for specific time. Due to the characteristics like determining quality and improving factor of software system, now a days it has not only been used in critical sectors but also in small organizations too. In the early phase of development process reliability prediction need to be done which provide quality enhancement, resource allocation for development and confidence to build quality software product. Based on the current scenario the software development process is gradually changed to component based development that increases the complexity of software. The present development process is suitable only for specific projects hence new development process using reliability prediction models is required to build to achieve failure free software. The main aim of this article is to suggest reliability prediction models to generate fault free software.**

Keywords*—Software Reliability, Software Failure, Reliability Prediction, Growth Model, Prediction Models

I. INTRODUCTION

The reliability defines approval or rejection of a software product. Practitioners try to enhance software reliability, so that they can minimize the development cost, economic competitions in the world [1]. Early phase of software development process is reliability prediction which offers quality improvement, resource allocation for development, testing purposes, which deliver outcomes on real time basis while quality of software determine by its reliability [2]. In other words, reliability prediction is helpful in acceptance of software product. It measures the capability of software to rectify failures. But some failures are irrevocable during software development process [3]. Developers should implement such features at the development process that can identify the error. at the initial stage. Further, basing on the criticality of identified errors, they are classified and try to mitigate such errors in management process [14]. For improving reliability of software such management process can be used to reliability prediction methods to mitigate the errors and faults.

The article focuses on reliability models and their role in real time systems [5]. Developers can use software reliability prediction model which provides a large information to estimate reliability[18]. A lot of growth models of software reliability are developed by different authors. But all growth models are not realizable except on particular data set or project [6]. Therefore, there is a search of such software reliability prediction model that suit to all data sets. The main task is to scrutinize the applicability of each software reliability prediction models that include features like reliability, concerning issues ideas for enhancement. This paper is totally focused on the reliability analysis models which are already existed. This can be applied in the real time software development process. All the reliability models are designed based on the some factor like Attributes, Design complexity, Reuse, Code Complexity, Post-delivery defects, Inspection, Execution operation, and Process and product metrics. There are two main types of software reliability models one is deterministic and another one is the probabilistic. The deterministic model is used to study the number of distinct operators and operands in a program as well as the number of errors and the number of machine instructions in the program. Performance measures of the deterministic type are obtained by analysing the program texture and do not involve any random event.

II. SOFTWARE RELIABILITY:MODELLING

The term software reliability defines the software will not arise any error in a particular environment for specific time period [7]. Additionally, the performance of the software is as expected in the pre-defined criteria without fail [9]. Quality of software is defined by evaluation of reliability. Software system is widely used by all organizations. In order to enhance the productivity of such system reliability is required. Reliability models provide information in respect of error detect, removal and details of environment that minimize the failure rate [10]. Hardware reliability could not be predicted easily with time while software reliability gradually improves throughout development process [8]. Software failures include software design, poor quality control, marketplace, capability profitable targets and engineering design estimation. Software reliability is a key part in software quality. The study of software reliability can be categorized into three parts: modelling, measurement and improvement. Software reliability modelling has matured to the point that meaningful results can be obtained by applying suitable models to the problem. Software reliability cannot be directly measured, so other related factors are measured to estimate software reliability and compare it among products. Software reliability improvement is hard. Realistic constraints of time and budget severely limits the effort put into software reliability improvement. All reliability modelling can be divided into two groups[11,12].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

- Black box software reliability models.
- White box software reliability models.

Black box models [13] applied to the different phases of current software development methods. This cannot be satisfying all the condition of component-based applications. White box models [14] may be satisfying all the condition of component based software. Now days it can be applied in the software development life cycle.

III. TYPES OF RELIABILITY MODELLING

Here already discussed Reliability modelling classified into two types Black box models, White box models Reliability models which is based on black box approach focus only the features of software's without knowing what happening inside knowledge of the structure. Black box modelling built on using failure data of system, several analytical models were proposed. Each model is categorized according to the fault data study given below.

A. Black box software reliability models

i. Time between failure models

This models relies upon the time (T) prepared between failure (n-1) and failure nth. Based on the value of time (T),required boundaries, reliability of software and interim to *next failure will be estimated by the fitted model Here, the* assessment depends on the quantity of shortcomings stay in the product during the interval time (T). Functional profile of the software and the quantity of lines of code ought to be reliable further- more, steady individually.

ii. The Jelinski-Moranda (J-M)Model

It is the best model [14] which is used for measurement of software reliability. This model is totally based on unfeasible assumptions. Because of its impracticable assumptions, this model can't cooperate in different data cases. The time interval between each failure is significantly distributed and unconventional. With a small time, all the ascertain faults are removed which does not create new faults during the execution process. The number of faults remaining in the software Is proportional to the failure rate of software and also it is constant in the failure interval.

iii. The Schick and wolverton (SW) model

This model[26] is similar to the Jelinski Morando model. Here the failure rate is calculated to be proportional to the actual fault material in the system, in addition to the time elapsed. Here the failure rate increases with time from the last execution at the nth time interval. In this model the failure rate of software R may be specified between (n-1) and nth failure.

iv. The Moranda geometric Poisson model

The Moranda geometric Poisson model (Moranda 1975) con- cludes the fixed times T, 2T, of equal length intervals and number of failures occurring at interval i, Ni, follows a Pois- son distribution. The model [14] consider to obey a Poisson distribution with intensity rate E mi-1 at fixed times T, 2 T, 3 T, 4 T . . . of the equivalent length interval, and the failures happening at an interlude i,ni. The process of calculating re- liability and other performance in J-M model is same in this model also.

v. The Goel and Okumoto imperfect debugging model and Non-homogeneous Poisson process (NHPP) model

Goel and Okumoto [8] proposed an defective model of executing by expanding the J-M model which, impertinent that a defect is taken away with likelihood p each time a failure arises. Models Goel and Okumoto [8] and J-M [24] trusted that the faults are suppress with cast-iron certainty when perceived but this won't be possible practically. It was expected to determine the imperfect debugging model. The faults in this model is at time t, Z(t), is preserved as Markov process where likelihood of imperfect debugging govern transition probabilities. Goel and Okumoto model NHPP model [24] assume that, the software put into failure activate by the faults remain in the system at random time. This paper also suggested the model to calculate the collected number of failuresduring time t.

vi. Fault seeding models

Fault seeding models is the strategy for embedding a known number of Fault or mistakes into the system where there is now an obscure number of shortcomings. In view of the additional or seeding issues into the program obscure number of issues will be determined. By utilizing these determined worth, required boundaries what's more, reliability analysis will be performed. The essential presumption of this issue or blunder seeding model are the seeding of the blunder or imperfection in the system follows arbitrary Distribution and Genuine and seeded faults have equivalent possibility of rev-elation .

vii. Input domain models

Sets of test cases are produced in input domain based models from an input distribution that is supposed to be representative of the program's operational uses. This class of models assesses a program or software's reliability when the test cases are randomly sampled from the well-known operational distribution of the input domain. The input domain is distributed in the equivalence classes group, each of which is associated with the direction of the programme. By finding all unique paths in the program and then execute each and every path is possible to guarantee that everything is tested. The basic assumption of this input domain model is Random testing has been used, partitioning the input domain into different groups and Known distribution of the input domain.

B. White box software reliability models

The key aim of White Box Reliability Models is to evaluate reliability by evaluating the internal coding structure and modular operating system interaction. This white box re- liability models also called as architecture based reliability analysis modelling. This architecture based reliability analysis modelling offers many advantage.These models of re- liability analysis allow one to compare the reliability of the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

application with its design and the reliability of individual components. By using this model's reliability of the application software will be measured early in the life cycle pro- cess. It enables us to correct the architectural design if there is any measured reliability. The intension of the architecture- focused software reliability analysis was to achieve a software reliability estimate based on the reliability of the componentsand software design [2,15].

i. Path based models

In a path-based model of reliability examination, the product's failure rate would be identified with the general paths execution recurrence. The software reliability is assessed by considering the conceivable execution paths of the program [12]. Numerous path based reliability analysis models have been proposed by scientists. The path based models expect that, every one of the segments are autonomous in nature. Failure of one segment won't influence another. Here programming design is joined with parts and interfaces. Un- wavering quality on the paths is controlled by increasing the dependability of the grouping of executed parts. Finally system reliability is estimated by averaging all the computed path reliabilities.

ii. State based models

Software architecture is addressed by the CFG (Control Flow Graph) in these classes of models. CFG is utilized to analyze the program code which has modules and choice focuses. This class of models expected to be that, move of control between modules has a Markov property which implies that the future conduct of the framework is restrictively free of the past conduct. The state based models used to address software architecture take record of a discrete-time Markov chain (DTMC), a continuous time Markov chain (CTMC), or a semi-Markov process.

iii. Additive models

This kind of model won't unequivocally think about the software system architecture. All things considered, they center on assessing the generally software reliability information from part or (node) failure. This model is additive since each part reliability is demonstrated by the Non-homogeneous Poisson process model. The amount of part failure intensity can be used to communicate the general sys- tem failure intensity.

iv. The Markov Structure Models

The property of this model is the future behavior of the process only depends on the current state and is independent of its past history. This Model is general way of representing the software failure process also it is used to study the interrelationship of the different modules and the reliability. It is also assumed the failures of the different modules are independent of each other. This expectation appear sensible at the mod- ule level as long as they can be designed, coded and tested separately but may not be true at the system level.

v. The Gokhale's model

This model[9] takes into account the time-dependent failure rates and the usage of the modules via the average estimated time spent in the module per run. In this method it provided via an absorbing Discrete-time markov chain. In this model it lies in the attempt of experimentally testing the application to determine component reliability analysis and software architecture. The upgraded non-homogeneous Poisson process model indicates the failure behavior of each variable using a time-dependent block coverage calculation of the failure intensity during application testing.

vi. The Vivek Goswamis's model

Vivek Goswamis's model [19] examine the reliability of software system by using discrete element reliability and the us- age of each element. The ratio of each element is calculated using the operational profile of software systems. Each element reliability is calculated by mathematical formula.

vii. Littlewood model

This is an earliest software reliability model [21] based on software architecture. It is trusted that an irreducible semi-Markov processes will define the software architecture while generalizing the earlier work expressing the software architecture with the Continuous Time Markov Chain. Here the software needed a proper finite number of modules and then control transfer between different modules. Every module when they are executed they loss their constant failure rates and also the component interface fail. This model is only validating using an imagined example.

viii. The H. Singh's Bayesian model

H. Singh method is used to imagine component based soft- ware system (CBS) reliability). This model [16] uses the Unified Modeling Language (UML) to derive the design of soft- ware system and its state that, reliability can be predicted in the system design level before starting the development. Using some case studies validation of this model is done. This model also guide examine the effect of replacing them with the more/less reliable ones and process of identifying critical Components. In the Bayesian estimation framework, posterior probability of failure is calculated from the test failure data and priors.

IV. IMPORTANCE OF SOFTWARE RELIABILITY PREDICTION

Everyone feels the need of software system in all fields that impact in each sector of the society [6]. This promotes the development of failure free software [8]. Such type of software applications requires labor intensive technique [5]. Hence it is required the development of reliable software that provide reliability accurately. The software failures have vital impact on economy as well as other factors. In order to avoid soft- ware failure the following parameters like error prevention, fault detection are checked. Currently different software metrics are available to measure the software and its reliability. Presently, a number of Software Reliability

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

Growth Models are found in the market but the prediction is not reached its highest. Higher reliability can be achieved by using better development process, risk management process, configuration management process, etc. Software failure is a common drawback in the present time as it could not reach the expected performance.

During each phase of software development, reliability factor is predicted. Plenty of growth models are developed for evaluation or operation phases while very less research is available for post-delivery phase. It is difficult to distinguish during developing software which effects more on reliability due to bug and defects. While measuring the reliability through the user experience, that will vary person to person, because each user experiencing different path or module of the software system [15]. Probability-based modeling has been suggested to resolve these issues to analyze and fore-cast the reliability of a software system with some hypothesis [1]. Each and every model is unique and based on the failure history of software, not a single model fit into all the cases. Every model has some advantages and limitations. Precise model needs to be chosen based on the software system, development process, input parameters available and assumptions to analyze the reliability of software System [10]. Still, many models are proposed by the researcher to assess the re- liability with limited assumption. Designing generic model is still a challenging issue in this research area [11].

V. SUGGESTIONS FOR PRACTITIONERS

During software development process errors are identified, detected and removed through software testing. The importance of preserving quality throughout software development has been examined and reviewed by Raksawat and Charoen- porn [17]. They addressed various software testing standards, such as ISO/IEC/IEEE 29119 ISO 29119, that can be used to conduct out software testing. The process of removing such error is called software reliability. The growth model of reliability is based on previous failures, which improves the quality of the software [1]. Reliability prediction determines faults and measurement of reliability. The reliability prediction is determined in the initial stage because it is difficult to examine at the end product. This article summarizes the following points.

- No thorough literature review of reliability modeling during development.
- Only a few tools of reliability modeling are merely used in real time systems.
- Almost all the models and methodologies have been implemented only on small and limited data sets. Hence need to conduct more experiments over large data sets and real world scenarios to extract concrete conclusion about the implication of software reliability growth models.
- This study emphasis on reliability prediction model which can produce reliable as well as usable software with the perspective of real time system.

Software Reliability is well-clearly said as a likelihood of the system functioning without failure for a definite duration in definite background [1] and also defined in such a way that, probability of software system or component will deliver its anticipated functionalities with excellence for a stated duration under the identified circumstances or software system deficiencies don't cause a fault during the predetermined time frame and condition. Modeling of reliability has been suggested to forecast and estimate the information systems re- liability. This modeling may useful in various stages of the software life cycle process [13]. Some modeling may be implemented in the initial development process and others may be implemented in the design or coding process, or in the test or repair phase. Each model has its advantages and limitations.

VI. SOFTWARE RELIABILITY GROWTH MODELS

The Software Reliability Growth Model[23,25] needed having a best performance in terms of goodness-of-fit, determinateness, and so away. In order to guess as well as to Forecast the reliability of software systems, failure data required to be accurately measured by different means during software development and operational phases. Any software needed to control reliably must still go through substantial testing and executing. This can be a expensive and time consuming pro- cess, and managers needs proper information about how soft- ware reliability developed as a result of this process in order to productively control their budgets and projects. The total effects of this process, by which it is expected software is made more reliable, can be matched through the use of Soft- ware Reliability Growth Models, here after consulted to as SRGMs. Research efforts in software reliability engineering have been guide over the past three decades and many soft- ware reliability growth models (SRGMs) have been put for- ward. These models contribute a means of describing the development process and enable software reliability specialist to make guess about the expected future reliability of soft- ware under development. Such techniques allow managers to exactly allocate time, money, and human resources to a project, and estimate when a piece of software has arrived a point where it can be released with some level of confidence in its reliability. Unfortunately, these models are often inaccurate.

i. Times between Failures Models

In this class of models, the interaction under examination is the time between failures. The most widely recognized methodology is to expect that the time between a dissemination whose boundaries rely upon the quantity of issues staying in the program during this span. Evaluations of the boundaries are gotten from the noticed upsides of times among failures and appraisals of programming dependability, mean chance to next failures, and so forth, are then acquired from the fitted model. Another methodology is to treat the failures as acknowledge of a stochastic cycle and utilize a fitting time- series model to portray the fundamental failures measure.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

ii. Fault Seeding Models

The fundamental methodology in this class of models is to ”seed” a known number of flaws in a program which is accepted to have an obscure number of native faults. The program is tried and the noticed quantities of seeded and native issues are checked. From these, a gauge of the shortcoming content of the program preceding seeding is gotten and used to evaluate software reliability and other pertinent measures.

iii. Input Domain Based Models

The fundamental methodology taken here is to create a bunch of test cases from an information appropriation which is thought to be delegate of the functional use of the program. In view of the trouble in getting this circulation, the info space is divided into a bunch of equality classes, every one of which is generally connected with a program path. A gauge of program reliability is gotten from the failures saw during physical or emblematic execution of the experiments tested from the input domain.

VII. Applicability of Software Reliability Models

We consider the four classes of Software Reliability Models and assess their applicability during the design, unit testing, integration testing, and operational phases of the software development process. During the design phase, faults might be distinguished outwardly or by other formal or casual methods. Existing software reliability models are not pertinent during this stage in light of the fact that the experiments expected to uncover. Blames as needed by fault seeding and input area based models don’t exist, and the failure history needed by time subordinate models isn’t accessible. During the unit testing phase, the ordinary climate during module coding and unit testing stage is to such an extent that the experiments produced from the module input space don’t frame an agent test of the functional use distribution. The time dependent models, particularly the time between failures models, don’t appear to be pertinent in this environment since the autonomous occasions between failures supposition that is genuinely violated. A regular environment during integration testing is that the modules are co- ordinated into incomplete or entire systems and experiments are created to check the accuracy of the incorporated frame- work. Test cases for this reason might be produced arbitrarily from an info conveyance or might be created deterministically utilizing a solid test procedure, the last being most likely more successful. The uncovered flaws are revised and there is a solid chance that the expulsion of uncovered deficiencies may present new blames.

VIII. A Software Reliability Modeling Example

We'll now use the technique outlined above to demonstrate how to create a software reliability model using failure data from a real-time command and control system. Bell Laboratories built this system, which had a total of 21 700 delivered object instructions. The problems observed during system testing over a period of 25 hours were reported by Musa [28], and they represent the failures encountered during system testing. The NHPP model of Goel and Komodo [29] is used in this example. We do so because of its simplicity and applicability across a wide range of testing scenarios, as indicated by Misra [30].

Step 1: The original data was presented as a count of how long it took for a failure to occur. We aggregated the data into numbers of failures per hour of execution time to overcome the potential lack of independence among these values. Table I summaries the information. Figure 1 shows a plot of the hourly data as well as a plot of N(t), the cumulative number of failures with time. A study of the data in Table II and of the plotting Fig. 2 indicates that the failure rate (number of The failure rate (number of failures per hour) appears to be decreasing with test time, according to the data in Table I and the charting in Fig. 2. As a result, an NHPP with a mean value function should be a suitable model for describing the failure process.

$$m(t) = a(1-e^{-bt})$$

Step 2: Two parameters, a and b must be determined from the failure data for the aforementioned model. For this, we choose to apply the greatest likelihood method [31],[32]. â = 142.32 and b = 0.1246 are the estimated values for the two parameters. Remember that is an estimate of the total number of faults likely to be discovered, whereas b is the number of problems found every hour.

Step 3: The fitted model based on the data of Table I and the parameters estimated in Step 2 is

$$m(t)=142.32(1-e^{-0.1246t})$$

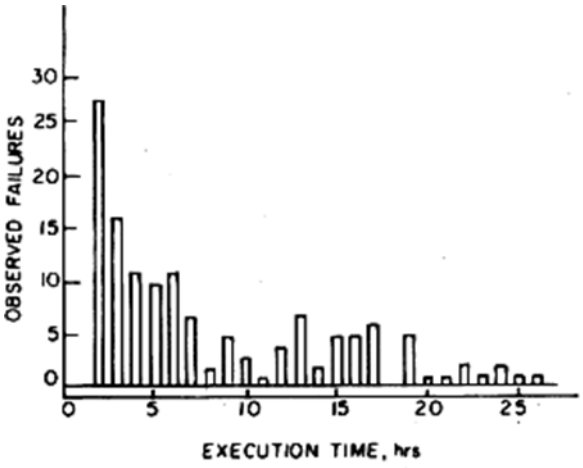

Fig 1: Plot of the number of failures per hour.

Table 1: Failures in 1 hour (execution time) intervals and cumulative failures

Hour	Number of failures	Cumulative failures
1	27	27
2	16	43
3	11	54
4	10	64
5	11	75
6	7	82
7	2	84
8	5	89

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

9	3	92
10	1	93
11	4	97
12	7	104
13	2	106
14	5	111
15	5	116
16	6	122
17	0	122
18	5	127
19	1	128
20	1	129
21	2	131
22	1	132
23	2	134
24	1	135
25	1	136

Step 5: To check the model's adequacy, we utilized the Kolmogorov-Smirnov goodness-of-fit test in this example. See Goal [33] for more information on this test. Essentially, the test is a statistical comparison of the actual data with the model chosen in Step 2. Step 3 of the fitted modeling passed this test, indicating that it is a fair description of the data in Table I. The plots in Fig. 2 also serve as a visual evaluation of the models goodness-of-fit.

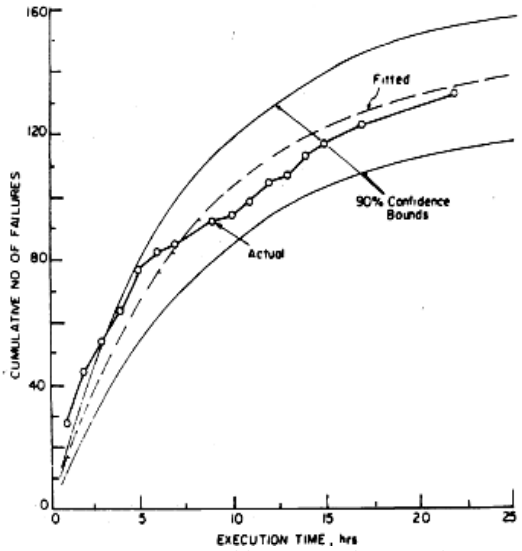

Fig 2: Cumulative number of failures as a function of execution time and confidence bounds.

Step 6: We computed only one performance metric, the predicted number of remaining errors, at varied testing times for demonstration purposes. Figure 3 depicts a plot of these values. Figures 2 and 3 provide plots of the confidence bounds for the estimated cumulative number of failures and the expected number of residual defects. 6 and 7 are the corresponding numbers. An examination of these plots reveals that the chosen NHPP model provides an excellent match to the data and may be used to describe failure behaviour as well as anticipate future failure processes.

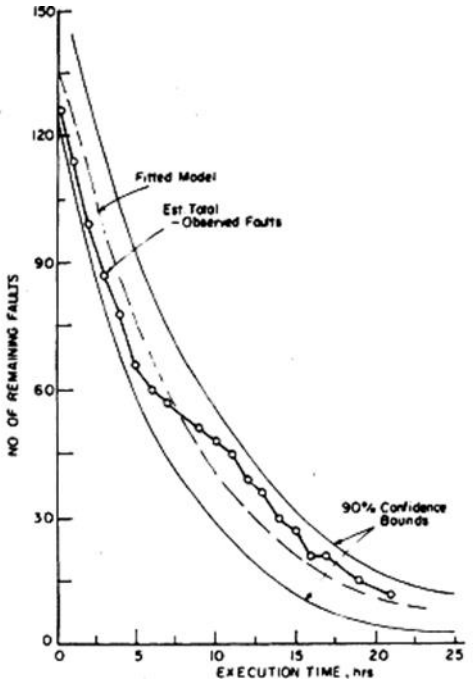

Fig 3: Estimated remaining number of faults and confidence bounds.

Step: 5 The model described above can be used to answer a range of questions regarding the failure process, as well as to determine how much more testing is needed until the system is ready for release. This type of information can be requested at any moment, and it is not necessary to wait until the completion of the testing process. Assume for the sake of illustration that failure data from only 16 hours of testing is available; with a total of 122 failures (see Table I). The fitted model is based on these facts

$$m(t) = 138.37\ (1\text{-}e^{-0.1332t})$$

Assume that the amount of residual defects is used to determine whether or not software should be released for operational usage. Assume we would release the system if the predicted number of remaining faults was less than or equal to ten. We can see from the preceding analysis that the best estimate of this quantity at this time is 16.37, which implies we should keep testing in the hopes of finding and removing more problems. If we repeated the analysis after each additional hour of testing, the estimated number of residual defects after 20 hours would be 9.85, allowing us to meet the stated release condition. The purpose of the foregoing basic example was to demonstrate the type of information that a software dependability model can provide.

IX. CONCLUSION

Software reliability is a concept that involves taking a step towards improvement and measurement of reliability. The reliability decides the acceptance or rejection of a software product. Generalized reliability prediction models are suggested by authors that can implement in each phase of development. This generalization will also reduce the time and cost spent on application of different reliability prediction tools at different phases. The main intension of this study is

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 09, September-2021

to energizing the young researchers to grow the new reliability survey frame- work which can be best and suitable for the modern software development methodology. Because all the developed models are having some pros and cons, due to this it can be suitable only to the specific projects and processes. The objective was to provide a user an insight into the usefulness of such models that will be helpful in determining which model to use in a given software development environment. At the time of acceptance testing, inputs based on functional usage are produced to verify software acceptability. In this phase, seeding of faults is not practical and the exposed faults are not usually immediately corrected. The fault spreading and times between failures models are thus not relevant. During the working phase, the user inputs may not be irregular. This is the reason user may use the same software function or path on a scheduled basis. In real-time systems Inputs may also be agreed, thus losing their randomness. Additionally, faults are not always instantly corrected. In this environment, fault- count models are likely to be most relevant and could be used for observing software failure rate or for determining the optimum time for installing a new release.

X. REFERENCES

[1] Kaswan KS, Choudhary S, Sharma K., Software Relia- bility Modeling using Soft Computing Techniques: Critical Review. J Inform Tech SoftwEng 5:144. 2015.

[2] C.Y. Huang, M.R. Lyu, Estimation and Analysis of Some Generalized Multiple Change-Point Software Reliabil- ity Models, IEEE Transaction on Reliability, Vol. 60, no. 2, pp. 498-514, 2011.

[3] J. E. Gaffney and J. Pietrolewiez. An Automated Model for Software Early Error Prediction. In Proc. of 13th Minnow Brook Workshop on Software Reliability, Blue Mountain Lake, NY,45-57, 1990.

[4] K. S. Trivedi, M. Malhotra . Markov Reward Ap- proach toPerformabilityand Reliability Analysis. In Proc. of International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, IEEE,7-11, 1994.

[5] J. D. Musa and K.Okumoto. Application of Basic and Logarithmic Poisson Execution Time Models in Soft- ware Reliability Measurement. Software System Design Methods, Springer, Berlin, Heidelberg, 275-298, 1986.

[6] V.Goswami and Y. B. Acharya. Method for Relia- bility Estimation of COTS Components based Software Systems. In International Symposium on Software Re- liability Engineering, 2009. [Online]. Available at: http://2009.issre.net/papers/issre2009 193.pdf

[7] M. Bisi and N. K. Goyal. Software Reliability Prediction using Neural Network with Encoded Input.International Journal of Computer Applications, 47(22),46-52, 2012.

[8] A. L. Goeland K. Okumoto. Time-Dependent Error- Detection Rate Model for Software Reliability and Other Performance Measures. IEEE Transactions on Reliability, 28(3), 206-211, 1979.

[9] S. S. Gokhale. Architecture-Based Software Reliability Analysis: Overview and Limitations. IEEE Transactions on Dependable and Secure Computing, 4(1), 32-40, 2007.

[10] M. Xie, 1991. Software reliability modelling (Vol. 1). World Scientific., ISBN 981-02-0640-2.

[11] M.R. Lyu. 1996. Handbook of software reliability engineering. IEEE Computer Society Press and McGraw- Hill., ISBN 0-07-039400-8.

[12] C.J. Hsu, C.Y. Huang, An adaptive reliability anal- ysis using path testing for complex component-based software systems, IEEE Trans. Reliab. 60 (1) (2011) 158–170, https://doi.org/10.1109/tr.2011.2104490

[13] S. Kaliraj, N. Chandru, A. Wahi. 2013. A Relia- bility Framework of Component Based Software System Using Kal-Chan Path Selection Algorithm. International Review on Computers and Software (IRECOS), 8(2), pp.605- 612. Available from: http://www.praiseworthyprize.org/.

[14] Z. Jelinski, P.B. Moranda. 1972. Software reliability research, Statistical Computer Performance Evaluation, W. Freiberger (ed.), 465–484. DOI: 10.1016/b978-0-12-266950- 7.50028-1.

[15] K. Gos˘eva-Popstojanova, K.S. Trivedi, Architecture- based approach to reliability assessment of software systems, Perform. Eval. 45 (2) (2001) 179– 204, https://doi.org/10.1016/s0166-5316(01)00034-7.

[16] K. Go, M. Hamill, 2007, July. Architecture-based software reliability: Why only a few parameters matter?. In Computer Software and Applications Conference.

[17] H. Singh, V. Cortellessa, B. Cukic, E. Gunel, V. Bharadwaj, 2001, November. A bayesian approach to reliability prediction and assessment of component based systems. In Software Reliability Engineering, 2001. ISSRE 2001. Proceedings. 12th International Symposium on (pp. 12-21). IEEE. DOI: 10.1109/issre.2001.989454.

[18] Chadatarn Raksawat and Pattama Charoenporn, ”Soft- ware Testing System Development Based on ISO 29119,” Journal of Advances in Information Technology, Vol. 12, No. 2, pp. 128-134, May 2021. doi: 10.12720/jait.12.2.128-134

[19] Seppo J. Sirkemaa, ”Key Perspectives in Informa- tion Technology Infrastructure Management,” Journal of Advances in Information Technology, Vol. 10, No. 3, pp. 100-103, August 2019. doi: 10.12720/jait.10.3.100-103

[20] V. Goswami, Y.B. Acharya. 2009. Method for re- liability estimation of COTS components based software systems. In Proceedings of 20th International Symposium on Software Reliability Engineering, ISSRE. Available from: https://www.researchgate.net.

[21] Y. Si, X. Yang, X. Wang, C. Huang, A.J. Kavs, 2010, April. An architecture-based reliability estimation framework through component composition mechanisms. In Computer Engineering and Technology (ICCET), 2010 2nd International Conference on (Vol. 2, pp. V2-165). IEEE. DOI: 10.1109/ iccet.2010.5485256.

[22] B. Littlewood, A reliability model for systems with Markov structure, Appl. Stat. (1975) 172–177, https://doi.org/10.2307/2346564.

[23] K.Sahu and R. K. Srivastava. Revisiting Software Reliability, Data Management, Analytics and Innova- tion.Advances in Intelligent Systems and Computing, Springer, 221-235, 2019.

[24] Mohd Razeef and Mohsin Nazir, “Reliability of Software Development using Open Source Technology”, in IJARCS, vol2, No. 5, pp. 479-485, Sept-Oct 2011

[25] A.L.Goel, “Software Reliability Models: Assump- tions, Limitations, and Applicability”, IEEE Trans. On Software Engineering, vol. Se-11, no. 12, pp. 1411-1423, December 1985.

[26] T. A. Thayer, M. Lipow, and E. C. Nelson, ”Soft- ware reliability study,” Rep. RADC-TR-76-238, Aug. 1976.

[27] G.J. Schick, R.W. Wolverton, 1973. Assessment of software reliability. In Vortra¨ge der Jahrestagung 1972 DGOR/Papers of the Annual Meeting 1972 (pp.395-422). Physica-Verlag HD. DOI: 10.1007/978-3-642-99746-430.

[28] J. D. Musa, "Software Reliability Data," DACS, RADC, New York, 1980.

[29] H. D. Mills, "On the statistical validation of computer programs, "IBM Federal Syst. Div., Gaithersburg, MD, Rep. 72-6015, 1972.

[30] P. N. Misra, "Software reliability analysis," IBM Syst. J., vol. 22,no. 3, pp. 262-270, 1983.

[31] A. L. Goel,"A software error detection model with applications," J.Syst. Software, vol. 1, pp. 243-249, 1980.

[32] A. L. Goel, "A guidebook for software reliability assessment," Rep. RADCTR-83-176, Aug. 1983.

[33] A. L. Goel, "Software reliability modellingand estimation techniques," Rep.RADC-TR-82-263, Oct. 1982.